A Cooperative Model for Integration in a Cardiology Outpatient Clinic

Ronald Cornet[1], Erik M. van Mulligen[1,2], Teun Timmers[1]
[1]Dept of Medical Informatics, Erasmus University Rotterdam, The Netherlands
[2]University Hospital Dijkzigt, Rotterdam, The Netherlands

ABSTRACT

With the increasing amount of digitally stored patient information, such as images and findings, the possibility and need arise for a system which is able to both store and display this information in a structured, user-friendly way. In the common situation where different information systems are used within one department, this means that the various information systems have to be integrated. However, integration requires more complex management of data, processes and windows. This management can be handled by a Workspace Manager, which controls inter-application tasks, and interaction with the user. Furthermore, this Workspace Manager supports the user with the definition of complex tasks such as collecting problem-related patient information from the various remote systems.

In an outpatient clinic for cardiology this architecture is used to achieve cooperative integration of an open Computer Patient Record with a range of information-specific services.

INTRODUCTION

Currently, much research is directed towards the development of a Computer Patient Record (CPR) [1,2]. In general, this leads to a stand-alone application that communicates only with various existing systems for data retrieval [3]. The disadvantage of such an approach is the lack of reusability of CPR functionality for different medical domains. Reusablity is highly desired for sharing of, e.g., modules for history taking, and display facilities for various kinds of signals, images, and data. Without reuse of components, a new CPR must be developed from scratch and no structural approach to improve reusability in an existing system is known. Another disadvantage of such an approach is the inability to easily integrate existing commercial solutions and new software developments within the CPR.

Other research in medical informatics has introduced reusable components as building blocks for a medical workstation, thereby using various mechanisms [4,5]. In this paper, we present an architecture that allows reuse of components in order to realize a CPR for a cardiology outpatient clinic. The functionality included in these components is either developed locally or supplied by commercial vendors. Beyond this, the architecture provides a powerful mechanism to anticipate new developments, such as digital image display, image analysis and clinical decision support.

In order to make a reusable-components based system act like a single application to the clinical end-user, the following architectural features are required (Figure 1):

(1) *standard communication* supporting the exchange of data and procedure calls between the components;
(2) *integration* supporting uniform access from one workstation to remote components as if they all reside on the same host;
(3) *cooperation* automating interaction between the components and serializing components for a user task;
(4) *user interfacing* for interaction with the user and presentation of data.

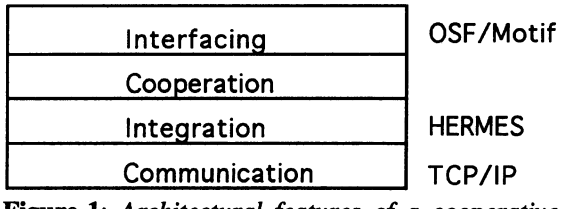

Figure 1: *Architectural features of a cooperative and interface-integrated system.*

Communication and integration are provided by the HERMES kernel architecture which was developed at the Department of Medical Informatics. This client-server-based kernel supports the integration of existing applications in a uniform way and provides a message language both for data exchange and remote procedure calling [6,7]. It promotes the development of reusable components as autonomous services in a network. The system runs on HP9000 /700 series workstations under UNIX with MOTIF. Together with the HERMES kernel, a set of basic

0195-4210/94/$5.00

services for access to commercial databases, to a Hospital Information System, and to a Departmental Information System is available. In a data model service additional information about the data is stored; this data model forms the bridge to a HERMES environment suited for clinical data analysis [8].

In this paper, we first explore the notion of cooperation as the next level of integration. Then preliminary results on the prototype of a Workstation for the cardiology outpatient clinic will be described.

COOPERATION: the next level of integration

On top of the HERMES integration layer, we have designed a new layer for cooperation. This layer, the Cardiological Workspace Manager (CWM), controls the operation of the components by means of the HERMES integration kernel. The reference model for this architecture is shown in Figure 2.

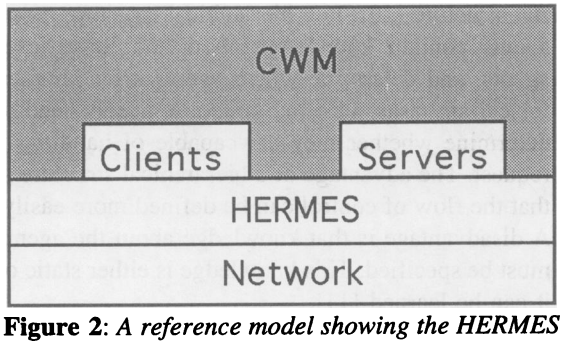

Figure 2: *A reference model showing the HERMES layers on top of the network.*

Requirements

The aim of a cooperative system such as our CWM is to support tasks that go beyond the scope of a single component, such as (1) decomposition of a user task into calls to the various components, (2) automation of the data- and command flow between components, and (3) a watchdog function to guard consistency and completeness of data.

Other requirements for the cooperative environment are its capability to include seamlessly new components, such as clinical decision support and literature access server components, and possibilities to modify and add tasks flexibly. The cooperative environment also has to provide functionality to anticipate multi-processing, i.e., to serialize and activate components concurrently that run on different hosts in a network with different response times.

This range of tasks requires an environment that supports the representation of procedural, event-driven and data-driven flows. In principle, one can choose from the following approaches to such an environment:

- *callback-based* (event-driven and procedural); environments using this mechanism associate events with callback procedures that on their turn can activate other callback procedures. When an event occurs, the system activates a callback automatically.
- *rule-based* (data-driven); in a rule-based system pre-conditions are expressed in terms of data. A rule can modify data, possibly causing other rules to be activated. As a consequence, one cannot implement a procedural flow.
- *object-based* (event-driven); each object is a combination of local data and methods. Methods can be activated by sending an event to an object, which in its turn can send events to other objects.
- *agent-based* (data-driven and procedural); an agent can be seen as an autonomous object; i.e. it contains a processor and incorporates its own data repository, logic and knowledge to reason about its environment. Methods can be called to manipulate data and delegate tasks to other agents.

As will be explained below, we have chosen for the agent-based model. The CWM has four classes of tasks:

User interaction tasks

The CWM provides the user-interface with a number of user tasks specific for the cardiology outpatient clinic. These user tasks include, e.g., support of consultation, review of patients, and generation of referral and discharge letters.

The CWM provides a mechanism to separate tasks in sequences of sub-tasks. Eventually, the sub-tasks open sessions with the appropriate services.

Window management tasks

A potential drawback of having many different components, each having its own windows for interaction with the user, is an uncoordinated appearance of the user interface. The CWM contains window management tasks that embed the server component windows in an overall user-interface that automatically positions, sizes, and (un)maps the windows.

Two modes for controlling the appearance of windows of the servers can be discerned:

- Reparenting; the window hierarchy (X11) is modified in such a way that a server window becomes a child of a window of the CWM. The window management tasks of the CWM operate through the parent of the server window.
- Communication; the CWM sends requests to the servers for positioning or hiding windows.

Which control mechanism is used depends on the server. Control by communication can be realized only for modifiable or new applications.
Display of the windows of the servers can happen in 3 different ways:

- A desktop metaphor; one window at a time is shown, the user can select windows by turning over "pages" of an imaginary notebook.
- Task-oriented window management: from a menu-bar, various types of information can be selected, and each source of information has its own area on the screen.
- Intelligent positioning: the CWM determines which information must be shown and where windows should be positioned.

Although flexibilty is decreased, window management is essential to avoid cluttering the screen with too many windows.

Inter-operation tasks
Since the CWM contains knowledge about the state of the servers, it can take care of a number of tasks considering information which is shared by the various servers.

- Clinical context: the CWM keeps track of the currently used domain and notifies the servers of the consequences of this domain.
- Synchronization:
  - of tasks: if a task consists of a number of steps, the CWM takes care of the correct sequence of sub-tasks.
  - of data: if another patient (or another global item) is selected by a server, the CWM notifies the other servers of the change.
  - of time frame: the CWM can handle a time-related user request, e.g. to display the first angiogram after the last PTCA.
- Communication: the CWM functions as a daemon to inform existing systems when data is stored or modified in one of the other systems [9]. For example, when an angiogram is stored in the angio-database, this is reported to the CPR.

Watchdog tasks
The data that is collected in the CPR will generally not only be used for patient care, but also for clinical data analysis. This brings up the need for a mechanism that checks data completeness and consistency. This will partly be done by the servers, but where the checks go beyond the scope of one application, this will be done by the CWM. Such a watchdog mechanism may also be used for educational purposes, to check if a user performs certain actions before making a decision.

WORKSTATION ARCHITECTURE

Cooperation mechanism
The mechanism we have chosen to achieve cooperation is the one of agents. The advantage of an agent-mechanism is that it is dynamic and that new agents can be incorporated easily into the system. This is in contrast with a rule-based or event-driven system, which leads to a static system with a large number of rules or triggers.
The agents can be organized in a hierarchical or a flat structure [10]. In a hierarchical structure meta-agents contain knowledge about the lower-level agents, and determine which agents are addressed for certain tasks. In a flat structure, agents need to determine whether they are capable of handling a request. The advantage of a hierarchical structure is that the flow of control can be defined more easily. A disadvantage is that knowledge about the agents must be specified. This knowledge is either static or it can be learned [11].

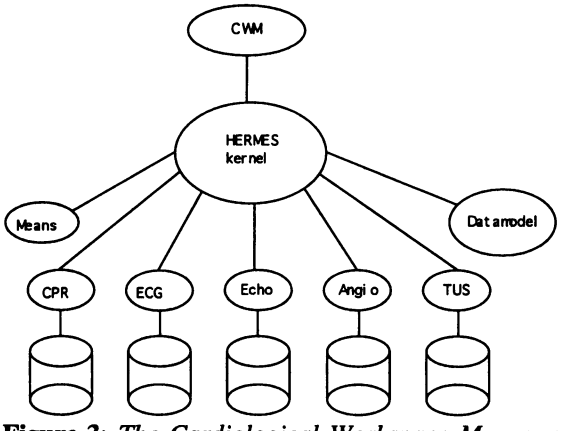

Figure 3: *The Cardiological Workspace Manager as a clinician's view to a number of information systems.*

Components for a cardiology outpatient clinic
Figure 3 shows the components of the cardiological workstation as it is currently developed. Other

servers which will be included are X-ray image and UMLS-servers, and a connection to Medline.

The following independent components will be connected in the near future.
Hermes Kernel: in use at various locations since September 1993.
Open CPR: The Computer Patient Record is adapted to the HERMES environment, according to the specifications as described in [1].
Angio: A CathLab database system has been developed during the last year.
ECG: The department of cardiology is currently considering which ECG-management system to use. Openness is a decisive requirement.
MEANS: An ECG-analysis module which has been developed at the department of Medical Informatics.
Echo: This year a VingMed image server on a SUN platform will be installed.
TUS: A MUMPS PC-networked Departmental Information System containing laboratory data, appointments, and other alphanumerical patient data.
Datamodel: The CPR consults this service for information on, e.g., the datatype, thesauri, and possible and plausible datavalues.

Local Database
Since all components are located at different hosts, the network load may be high, decreasing the speed

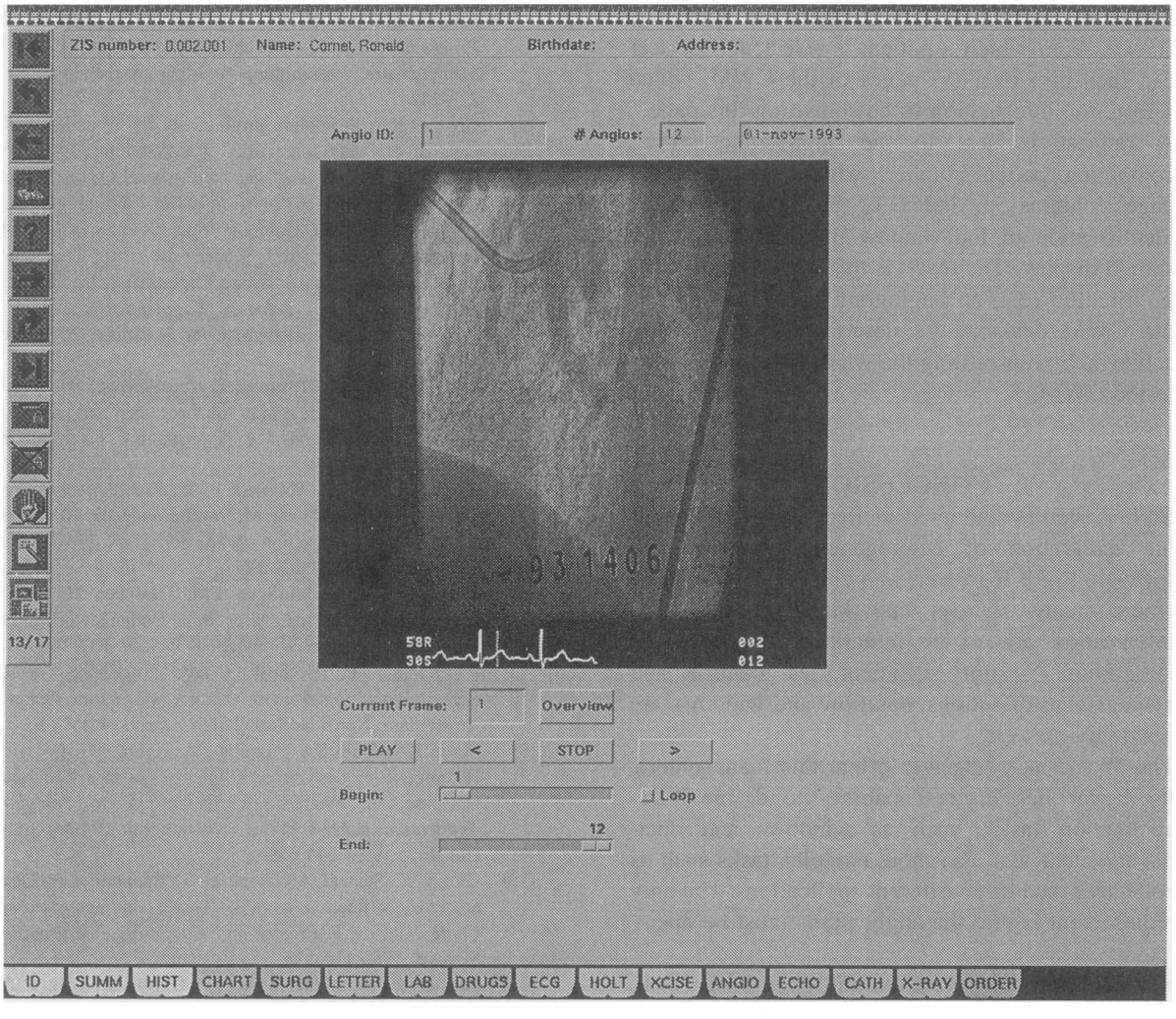

Figure 4: *An example of integrating the interface of a server into the Cardiological Workspace Manager: an angiogram is displayed as one of the pages of a notebook, being the desktop metaphor of the Patient Record. The user can select the type of information through the tabs at the bottom of the screen. With the buttons on the left hand side the user can navigate through the "notebook".*

of the system. Therefore, the use of a local database must be considered for storage of pre-fetched data. Agents, being active in the background in a transparent way, may provide an ideal solution to the problem of synchronization of the information in remote components. In particular, an agent can generate alerts where an existing, not open legacy system is not able to do that.

Sample screen

Figure 4 shows a sample screen of the prototype workstation. Currently, the following server components have been included: appointments, vital signs, letters, laboratory-data, ECG-signals with interpretation, angiograms, and echograms.
When the CWM is started, connections are opened with each individual server. Then, the agenda is shown, from which a patient is selected. The user can then turn over the pages of the CWM, which selects the information to be shown on this page.
A request is sent to the appropriate server, containing patient-number, whether the selected page contains an index or an image, and an identification of the window in which the server should display. The result is that the user views the information (e.g. an angiogram) as if it is part of the CWM desktop. As described above, this is achieved by reparenting the windows of the servers to the CWM.

CONCLUSION

Earlier, we have adopted a client-server model for the integration of heterogeneous systems and software [6,7]. However, in a clinical environment, where many sources and kinds of data and information are to be integrated, this level of integration is not sufficient, as control and management of actions, dataflow and windows are necessary as well.
The Workspace Manager offers this management. Its power lies in its capability to define inter-application tasks, such as window- and data-management, but also more complex tasks such as collecting data from different applications. The only requirement is that the applications must be open.

Several important issues are still open to research. One issue is the granularity of the agents and how their functionality is defined [12].
The proper balance must be found between pre-structuring user-interfaces and allowing maximum flexibility. Related to this is the question of how and to what extent one can tailor the behaviour of the interface to a particular user.
Finally, the pervasive problem of correctly mapping data in existing systems into the CPR remains. The additional features of the datamodel service may play an important role in this.

References

[1] Ginneken AM van, Stam H, Duisterhout JS. A Powerful Macro-model for the Computer Patient Record. Accepted for *Symposium on Computer Applications in Medical Care*. Washington, DC: 1994.

[2] McDonald CJ, Barnett GO. Medical record systems. In: Shortliffe EH, Perreault LE, eds. Medical Informatics: Computer Applications in Health Care. Reading MA: Addison-Wesley, 1990:181-218.

[3] Suermondt HJ, Tang PC, Strong PC, Young CY, Annevelink J. Automated Identification of Relevant Patient Information in a Physician's Workstation. In: *Proceedings Symposium on Computer Applications in Medical Care*. Washington, DC: McGraw-Hill, Inc., 1993: 229-232

[4] Greenes RA. Promoting productivity by propagating the practice of "plug-compatible" programming. In: Miller RA, ed. *Proceedings of the 14th Annual Symposium on Computer Applications in Medical Care*. Washington DC. Los Alamitos: IEEE Computer Society Press, 1990:22-6

[5] Degoulet FJ, Coignard J, Scherrer JR et al.. The Helios European Project on Software Engineering. In: Timmers T, Blum BI (eds.) *Software Engineering in Medical Informatics*. Amsterdam: Elsevier Scientific Publishers. 1991:125-37

[6] Van Mulligen EM, Timmers T, Van Bemmel JH. A new architecture for integration of heterogeneous software components. Methods of Information in Medicine 1993;32:292-301

[7] Van Mulligen EM, Timmers T, Brand J, Cornet R, Van den Heuvel F, Kalshoven M, Van Bemmel JH. HERMES: a health care workstation architecture. J of Bio-Medical Computing 1994;34:267-275

[8] Timmers T, Van Mulligen EM, Van den Heuvel F. Integrating clinical databases in a medical workstation using knowledge-based modeling. In: Lun KC, Degoulet P, Piemme TE, Rienhoff O, eds. *Proceedings of the Seventh World Congress on Medical Informatics*. Geneva. Amsterdam: North-Holland Publ Comp, 1992:478-82

[9] Marrs KA, Steib SA, Abrams CA, Kahn MG. Unifying Heterogeneous Distributed Clinical Data in a Relational Database. In: *Proceedings Symposium on Computer Applications in Medical Care*. Washington, DC: McGraw-Hill, Inc., 1993: 644-648

[10] Giroux S, Senteni A, Lapalme G. Adaptation in Open Systems. Reflection as a backbone. In: *International Conference on Intelligent and Cooperative Information Systems*, 1993:114-123

[11] Namatame A, Tsukamoto Y. Learning Agents for Cooperative Hyperinformation Systems. In: *International Conference on Intelligent and Cooperative Information Systems*, 1993:124-133

[12] Mylopoulos J, Rose T, Woo C. Task-Oriented Development of Intelligent Information Systems. In: *International Conference on Intelligent and Cooperative Information Systems*, 1993:206-219